BENCH TEST

ACORN ATOM

The last year has spawned an ever-growing number of single board computers that offer, in assorted shapes and sizes, the ability to program relatively easily in a high level language and at a reasonable price. The Acorn Atom is part of this group and its recent launch has been accompanied by some very attractive advertisements. In this first-ever Bench Test, Mike Dennis reveals all there is to see of the Atom's totally British design.

Introduction

It only seems yesterday that the original Acorn was launched and since that date many units based on Eurocard size boards have been announced by Acorn to augment the range. The Atom, however, is a departure from this philosophy in so far as it's a self-contained computer in its own right with its own expansion 'sub-set'. This ranges from the basic kit at £120 to a fully expanded Atom for £250; in other words it's designed to appeal to both wide tastes and, perhaps, not so wide pockets.

Hardware

The Atom is packed in a small polystyrene box of quite small dimensions — 15" x 10" x 3". The keyboard is integrally mounted on a sloping front panel and sports a full complement of 60 keys; at about 7lbs the whole issue is surprisingly compact and light in weight. It's in fact quite similar to the TRS-80 in dimensions and suffers from the same drawback of needing an external power supply — something extra to cart around when you have to clear the kitchen table! The overall impression left me cold and disappointed and having seen the superlative photographs in the glossy adverts it merely confirms my opinion that photogenic really means 'looks better on film than in real life'.

The minimum connections required to the outside world are a DC supply of capacity dependent on the system configuration and, for once, a decent professional BNC video socket instead of those abhorrent phono plugs. Hopefully, the production models of the Atom will mount this socket rather more substantially than on the prototype. Program storage is on cassette and a seven-pin DIN plug is provided for that. Further expansion is facilitated by bringing out various bus lines and other signals to a 64-pin Eurocard type connector (one that's a standard feature of all Acorn products). However, before you can use this facility you have to install some additional buffer components inside (the sockets are already there) — but that's not mentioned in the sales blurb. The DC input plug is of a relatively obscure design and hopefully Acorn will supply suitable leads both for this and for the video.

The Atom is available in either a kit or an assembled version; the model I had for review was a production prototype and therefore already assembled. I can't foresee any difficulty in construction as the component density is not particularly high. Unfortunately, the assembly instructions weren't available at the time of this test.

The minimum system consists of the 8k Basic in ROM and 2k of RAM giving a cost of £120 in kit form or £150 assembled. Of the available 2k of RAM, 512 bytes are used for the screen memory and 1k for the system, leaving only 512 bytes available for programs. That's not a lot of room, but a further 10k of static RAM (2114s) can be added to the Atom giving a total on-board RAM capacity of 12k. The price of 1k of RAM from Acorn is £9.50 and so you would do well to look through the adverts to compare prices. The other point regarding this extra expansion is that it doesn't all appear to be available for program space as a contiguous block.

The memory map is a bit weird and depends on the system configuration — See Fig 1. The unexpanded 2k Atom uses the 512 bytes from 8200H to 8400H as program space but when any expansion takes place (from 2800 upwards), the Basic sniffs out this new RAM and uses that to store programs instead. This in turn releases the previously used 512 bytes at 8200H for use by the system's high resolution graphics. There are thus two main areas of RAM

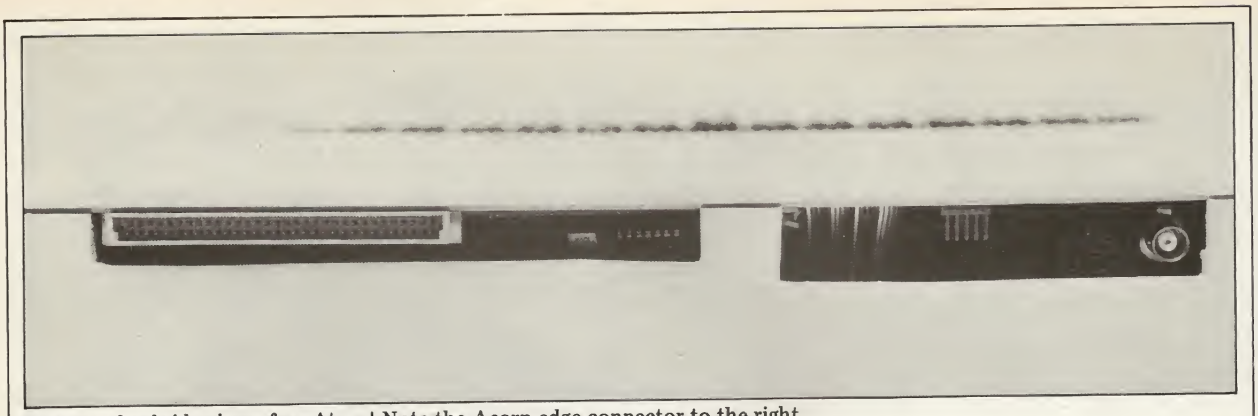
Unexpanded	Expanded	
0000 — 0400	0000 — 0400	System RAM
8200 — 83FF	2800 — 3C00	Program space
8000 — 81FF	8000 — 9800	Graphics space
A000 — AFFF	A000 — AFFF	Spare ROM space
C000 — CFFF	C000 — CFFF	BASIC
	D000 — DFFF	Floating point
F000 — FFFF	F000 — FFFF	Operating system

Simple, neat, perhaps a little plain in outline. . . the Acorn Atom.

in the fully expanded system, a 5k block from 2800H to 3C00H and a 6k graphics memory from 8000H to 9800H, and at first sight you can't use any of the graphics memory for program storage. However, buried in the manual is a reference to shifting the program area; changing a byte at location 18H from 29H to 82H will start program execution at address 8200H. This is a very good method of effectively providing a pseudo-turnkey system as you can use a spare 4k slot at A000H to install a ROM with some Basic program stored in it, change the data at 18H to A0H and run the program. There's no reason why you couldn't have several Basic programs in RAM and execute whichever one you choose merely by changing this byte. The one drawback that I see — and cannot find any easy solution to — is how to combine the two areas of memory together into one effectively continuous block for very long programs. Another fact is that the full 6k of graphics memory is required for high resolution graphics . . . a point that is made none too clearly in Acorn's adverts.

The existing firmware can be expanded by a 4k ROM with all the floating-point handling routines, and colour routines — again, a fact that the adverts don't make particularly clear. However, this ROM costs only an extra £20, which is good value for money. The last aspect associated with expansion is the question of the DC supply required. The Atom technical description specifies 8 V at 800 mA for an unexpanded Atom and there's an internal regulator. A fully expanded system needs +5 V at 1.8 A from an external supply and, presumably, the internal regulator is not moved — the manual isn't clear.

The remaining major chips in the system are the 6502 processor, an optional 6522 VIA (Versatile Interface Adaptor) that, incidentally, is needed in order to interface a printer, an 8255 that I presume looks after the keyboard and a 6847 video display generator to handle the screen video and graphics. Any data sheet on the 6847 will indicate that all the various clever graphic modes that the Atom offers are in fact nearly all done for you by this chip. But it suffers from the disadvantage of being designed for the NTSC standard and so generates a 60 Hz field rate as opposed to our own 50 Hz. The upshot of this is that you will almost certainly have to adjust the vertical hold of your TV set. This is of no great importance if the control has sufficient range but a lot of TV sets that apparently lock satisfactorily are in fact near the limit of their range with the result that some slight vertical jitter may be visible. Also, many TV sets have this control mounted inside the cabinet and so frequent tweaking to watch 'Crossroads' is out. I would also like to have seen to what degree of success Acorn has managed to produce a decent colour picture from this chip, but unfortunately my Atom didn't have the



First ever backside view of an Atom! Note the Acorn edge connector to the right.

necessary option.

The final chip (in the review sample) was a 68B54 ADLC or Advanced Data Link Controller; it's installed in the Atom as an optional extra when the Atom is to take part in the 'Cambridge Ring'. No, this isn't East Anglia's answer to Bayreuth but a communications concept that links several computers together.

After switch on, if you follow the instructions in the manual, then you may never get a picture because sometimes there is a totally meaningless jumble that no TV set will lock on to in a month of Sundays. At a guess, I'd say that the initialisation routine of the VDG wasn't always happening — possibly as a result of a power-on reset not always working. Hitting BREAK is the answer whereupon 'ACORN ATOM', a prompt and a cursor appear on the screen. The display format, a rather weedy 32 x 16, was effectively fixed the day Acorn decided to use the 6847 VDG. The chosen character set is fairly standard and lives inside 6847, but without lower case. It also uses the square 'O' that I personally think looks wrong on the screen.

The keyboard has a number of useful features like direct cursor control and a copy key. This is extremely useful as it simplifies correction of programs; you can shift the cursor to the beginning of the errant line, press COPY and REPEAT together, whereupon the cursor will beetle along the line and effectively re-enter it into the program store. When you reach the erroneous code in the line, you simply retype it and then press COPY and REPEAT again to finish entering the line. If you've ever used the Apple then you'll know what I mean. All the keys themselves have a very silky feel to their action . . . a pity then on my machine that several failed to operate reliably unless given a good thump.

Another aspect of the display that's potentially quite good is the fact that all numbers are right justified with an eight digit character field. Unfortunately, the 32 characters per line means that the maximum number of columns you can have in a line is limited to only four. The number of characters per field can be altered with the @ key but I found that altering it from the standard value of eight generally detracted from the overall legibility of the display. Unfortunately there is no TAB function. I found the legend on the @ key rather amusing as it appeared to have been made from two Letraset characters . . . I hope Acorn are getting some engraved keys made! The other feature of the display is the fact that carriage returns hardly ever happen and

```
10 PRINT "123"
20 PRINT "ABC"
```

will result in a display of "123ABC" — the first of many inconsistencies with standard Basic. Since carriage returns seem few and far between, the prompt flits about the screen and you're never quite sure exactly where it will pop up next.

Reverse or inverse video and certain double-function keys are accessed with the aid of either SHIFT lock or the normal SHIFT keys; however the positioning of the SHIFT LOCK key could have been improved as it was too easily to hit it when aiming for SHIFT. Since SHIFT LOCK has a toggle action, quite a few inverse characters appeared!

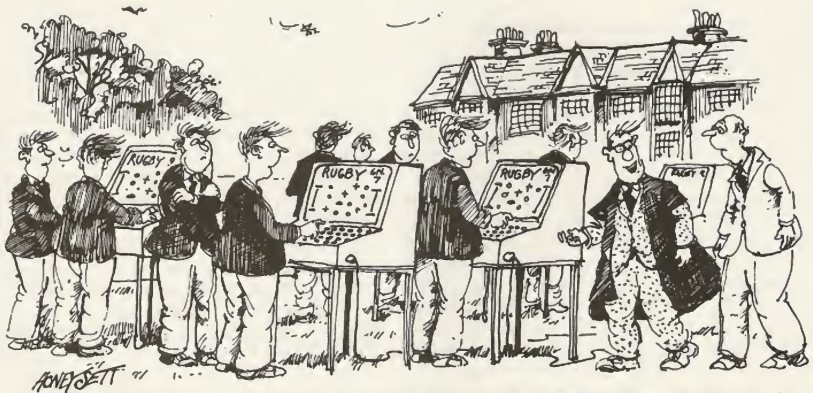
Program or data storage on cassette is quite versatile and I shall go into that in greater detail later on. Fairly comprehensive setting-up routines are provided to optimise the cassette recorder volume control, although the manual isn't too specific as to whether it means playback or record; actually, they do mean playback. The replay side of the Atom is insensitive as my own recorder pushes out quite a few dB of signal and should have been more than enough to drive the Atom. I found I had to drive the tape well into overload in order to get sufficient signal to load programs reliably. Acorn is not alone in this aspect and it's about time computer manufacturers who rely on cassettes for storage woke up to the fact that there's a wide range of sensitivity and output level among cassette recorders. The input side of any computer needs an adjustable high gain to cater for this and a test-tape to be supplied with each computer.

Other system commands include a comprehensive LISTing facility, albeit at a relatively slow scroll and display speed. There is no AUTO line numbering, nor RENUMBER, although a program to do this is given in the manual — a bit fiddly. One joke in the manual is section 1.7 Editing: "One powerful feature of the Atom's text and program storage is that stored lines can be modified very simply by typing the same line number followed by the new version . . . To delete a line, simply type in the line number followed by return . . . " WOW! In fact, there is no delete facility so you'll have to enter an awful lot of line numbers. That's hardly what I'd call 'powerful editing features'.

Software

The minimum system comes complete with an 8k Atom Basic — which includes an assembler and operating system within that space. As can be seen from the memory map, it's split into two 4k blocks. Firmware expansion is, as I said earlier, a simple matter of plugging a 4k EPROM into slot D000H.

As can be seen from the Benchmarks, Acorn's claim for high speed execution is certainly met, at least when integer Basic is used. Table II shows the actual timings. A closer examination of the individual programs shows that from Benchmark 3 onwards some of the routines will cause rounding up to occur by integer Basic. The timings, when repeated using the floating point routines, were much slower and not particularly exciting. However, the most interesting fact to emerge from running these programs was my initial difficulty in getting some



"The headmaster's very keen on new technology, but he still believes in the good old fashioned virtues of outdoor sport."

```
1000h IF?D=0 RETURN
1010  D!4=!D—1;D?6=D?1;D?5=D?2;D-D+4;GOSUBh
1020  MOVE(F—D?—4+D?V*N—N), (D?V?A*2);PLOT1, (D?—4*2—1),0
```

Figure 2

of them to run at all. Normally I run them without any problem so you can imagine my surprise when some of them started giving error messages. A trip into the manual solved the immediate problem but fundamentally revealed that Atom Basic is non-standard in many, many ways.

There are only 27 integer numerical variables allowed (@ to Z), stored in four bytes each with 32-bit precision. The maximum figure quoted that they can represent is 2,000,000,000 but in reality 2,147,483,647 is the true maximum — which will give you some idea as to how they are stored. This is a distinct improvement on most integer Basics with their maximum of 32,767, but 27 variables is not very many. You can't have A1,A2 etc. None of the variables are cleared at RUN time which can cause problems unless you rigidly define them in your program. These variables are then built on by appending seven different characters before the variable letter. These are S,?,!,#,&,% in floating point and the letter again in arrays — which makes program listings unnecessarily complicated (see the listing in Fig.2, taken from the manual). Other non-standard features became apparent the more I delved into all the variable options. Note that string variables are $A and not A$, that it's LEN(A) and not LEN(A$), that you must DIM all string variables before you use them and that it's DIMA(10) and not DIM $A(10) and that you can only have 27 string variables. Well, that's not quite true because there's a bit of a faux-pas in the Basic in that you can't have the same variable as an integer variable *and* as a string variable; if you use up all 27 variables as numbers then you can't have any strings! Fortunately you can have floating point variables which don't have the same effect, but even so it's a bit of a drop-off.

The ? prefix is a substitute for PEEK and POKE and the function it performs depends on the context in which it is used:
PRINT?A is the same as PRINT PEEK A
while
?A=13 is the same as POKE A,13.
The manual describes this as "a more elegant way" but I leave it for you to decide which is the easier to comprehend especially as ? crops up again but this time in the string section for:
10 $A="MIKE DENNIS ASSOCIATES"
20 PRINT A?3
will print the E from MIKE. Apparently you can use PRINT3?A instead, which will give the same result. Confused?

The ! or 'pling' as the manual calls it is used in 'word arrays' and stores numeric variables in four-byte chunks. The advantage of these over the normal array is that the vector to each array can be passed as a variable to a subroutine — which is quite useful. Remember ?, well it can also be used to store 'byte-vectors' which stores elements in one-byte chunks — all great stuff.

Two of the remaining prefixes,# and &, are used in hex to decimal conversion and the idea is very sound. Machine codes and addresses are in hex: Basic considers everything as a decimal and you have to convert between the two either by hand or with a Texas Programmer.

The Atom cheerfully accepts either format with these two prefixes and converts between the two for extra convenience. For example:
PRINT # 8000 will print 32768 and
PRINT&32768 will print 8000;
PRINT& # 8000 will print it in hex!!
This is a very worthwhile feature of the Atom. Unfortunately, the last prefix, %, has two different functions, depending on whether it refers to integer or floating point. That's not strictly true, because in integer you'll find it between two variables whereas in floating point it will occur before — but it *is* easy to get confused.

That means a total of six possible prefixes before a variable, which makes program comprehension (not to mention portability) very, very difficult. I'd also have thought that all these prefixes made it unsuitable for teaching. Now perhaps I'm being a bit reactionary; if I'd never neard of Basic and perhaps had been programming in 'Softo' for a long time — then someone introduced me to the Atom — well, I just might have leapt up and down and thought it the best thing since sliced bread. But Basic per se has been around for a long time and with relatively uncomplicated variables (thems with $s and thems without). As a consequence, the Atom seems more and more like an ego trip for a Cambridge don.

Arrays are also limited to one-dimension. This time 26 arrays can be used and they are accessed by prefixing each variable with itself, i.e. AA. Multi-dimensional arrays are out although Acorn does provide a fudge routine to get round this problem; but surely the point is that these days you shouldn't have to.

I've already mentioned the position of the $ sign in strings and other differences as well. CH"A" will print the decimal equivalent of the ASCII code for A and should not be confused with CHR$(A) in normal Basic for its real equivalent is ASC(A$). CHR$ is not provided and neither is SPC — so formatting tables will be a real pig.
Concatenation is $A+LEN(A)=$B and not A$+B$. Substring handling is:
$A+N equivalent to RIGHT$ (A$,N)
$A+N=" " equivalent to LEFT$ (A$N)
Two lines of program are needed to do a MID$. One-dimensional string arrays are available but you have to write a short program to dimension them, unlike other computers which dynamically alter the string array space to suit each appropriate string. Neither READ nor DATA statements are provided and, yes, you've guessed it, Acorn supplies a program that you have to type in to perform what should normally be a simple function. I hope you're following all this because "Atom Basic . . . has all the normal functions you would expect plus many other powerful extensions making it easier for you to operate . . . " I offer you a short excerpt from a program given in the manual and leave the final decision on the ease of programming to you (see Fig.2 again).

There are other inconsistencies such as ; to delimit multiple statements on a line instead of the usual : (that's used in the Exclusive-OR statement) but perhaps I ought to talk about the good points.

The best of these is that you can enter assembler mnemonics as program lines within a Basic program and when the program is run, the assembler part will be assembled and an assembler listing printed out. Should you so wish, the program will jump to the start of the assembled program with the LINK command and a return to other parts of the Basic program can be made with equal ease. Any parameters that Define Byte and Define Word in a normal assembler are set up prior to assembly by other lines in the Basic program and passed as variables to the assembler. Labels are allowed but if a reference to a label is made before it has been defined then an Out of Range error is flagged that can be fixed by RUNning again. Although this is often handled automatically by normal assemblers it is of no great disadvantage to have to enter RUN twice with the Atom. The real power of this assembler lies in its being interactive with Basic and therefore being able to use all the power of Basic conditional branching — which means that the Atom can handle conditional assembly with ease. Macros can be handled just as easily, particularly as another good feature of the Basic is that GOSUBs can be referenced to a label, i.e:
1000s (LSR A; LSR A; LSR A; LSR LSR A)
1010 RETURN
can be called by the simple expedient of GOSUB s. The only real criticism of the assembler that I could find was the inability to comment on individual lines. It even supports breakpoints!

Other interesting features are the DO . . . UNTIL loop, which is fairly obvious, and WAIT; this command waits for the next vertical sync pulse in the VDG and so provides an accurately timed reference point with which to synchronise. It's not just a simple time delay of 17 ms as the overall delay would then depend on the time taken to execute other instructions. WAIT is better than that and, used as an element in a timing loop, it effectively pads out the loop to exactly 17 ms — which is quite a useful feature.

Data storage on cassette is quite powerful and there are several commands that can put all the different variables to tape and load them back. One important point mentioned in the manual is that when data is saved on a cassette then the program reading the data back must not take any longer than the time taken for the initial saving, otherwise bits and bytes will come off the tape and be missed by the program. This is inherent in any cassette system that doesn't have its tape transport mechanism under program control and is not intended to be a criticism of the Atom. In fact I would say that the Atom is one of the few single-board computers that allows you to create a decent data-base on cassette.

The other good point is that all the program commands are designed to be compatible with any future disk system, thus obviating the need to rewrite any

BASIC COMMANDS

ABS	AND	BGET	BPUT	CH
CLEAR	COUNT	DIM	DRAW	END
EXT	FIN	FOR...NEXT		FOUT
GET	GOSUB	GOTO	IF	INPUT
LEN	LET	LINK	LIST	LOAD
MOVE	NEW	OLD	OR	PLOT
PRINT	PTR	PUT	REM	RETURN
RND	RUN	SAVE	SGET	SHUT
SPUT	STEP	THEN	TO	TOP
DO...UNTIL		WAIT		

$ % & # ! ? : '

FLOATING POINT COMMANDS

FIF	FINPUT	FPRINT	FUNTIL	STR
FPUT	=	!	ABS	ACS
ASN	ATN	COS	EXP	FLT
HTN	LOG	SIN	SQR	TAN
VAL				

programs when you upgrade.

The floating point routines are also interesting in their own right as they feature ARCSIN, ARCCOS and ARCTAN — which is quite unusual for a small computer. The accuracy of the floating point package is also exceptionally good and it's a pity it doesn't support degrees in addition to radians. Other facets of this package have been discussed elsewhere and I shall now turn to the graphics section.

There are many options available to the user, ranging from coarse graphics with mixed text and a low memory overhead through to high resolution graphics with no mixed text but a high memory overhead. Commands are available to move, plot and draw either black lines on white, white on black or even to produce a form of three-tone picture with a mid-grey option. The actual line movement can be done relative to the last position, or some other references, and the whole facility is very comprehensive. The only criticism is the fact that in the high resolution mode, the effective display area shrinks noticeably to about two-thirds of its previous size and a crude white frame of chunky proportions surrounds it which detracts from the overall effect.

Expandability

Apart from the internal RAM there is the further option of connecting up any of the other boards in the Acorn range. However you would be well advised to check with Acorn with regard to loading of the lines etc and also *exactly* what extras are needed in the way of buffer chips. Another fact that you should be aware of is that, due to the peculiar memory map, it's possible that some of an 8k static memory board will not be 'seen' by the Atom as it conflicts with the Atom's memory which takes precedence. This is even more of a problem with the 32k dynamic RAM board that they supply, for only 23k is actually available on the Atom, so wasting 9k, which is ludicrous. Hardware expandability is therefore reasonable but not as good as it could have been if it weren't for that weird map.

BENCHMARKS

	Integer	Floating Point
BM1	0.8	
BM2	5.5	
BM3	10.0	30.5
BM4	11.5	27.0
BM5	14.5	30.00
BM6	20.0	
BM7	31.5	
BM8		26.0

Documention

The Atom comes with a very chunky book that starts off extremely well. It's billed as a beginner's course in Basic and machine code programming and generally it's very good and gives plenty of examples. However a lot of the programs could be better documented and a greater variety offered. There are far too many mathematical programs that will leave the average punter cold; he doesn't want to know about Sierpinski curves and to put the following sentence in a beginners' book is just plain stupid: "This method has the advantage over the standard pivotal condensation technique that for integer coefficients the answers are exact integers or fractions". What did I say about an ego trip for a Cambridge don? There are no page numbers and without the index it's very difficult to find your way around. Also, there are several programs which use commands that haven't been explained before and therefore are very confusing. The programming analogy of making a cake is super and makes 'yer pivotals' look even more soppy. Error handling codes are available at the back and although they aren't all there (I got 89, which doesn't exist according to the book) this is in hand.

Generally, though, despite the maths overkill, a good effort which must have taken a long time to prepare. The technical manual is under preparation and if the standard of Acorn's documentation is anything to go by, then it should be good.

Potential use

Clearly the Atom has been aimed at the educational market. Its business applications are severely limited and I would like to see someone actually try and use it for some of the applications that Acorn suggests in its sales blurb. I can see it going into the educational field partly because of the 'Ring' and partly because on paper it seems to offer a lot of facilities for the money — which will probably prove attractive to impecunious education authorities. Personally, I think that as a beginner's tool it has far too many features to aid clear thinking and comprehension.

Conclusion

The Atom certainly offers many facilities; in particular, the combined assembler and Basic is particularly attractive. However, the very extent of these features makes it unwieldy to use at times — a fact borne out by the difficulty in following many of the program examples given in the manual. The incompatibility with virtually every other Basic (it really does seem as if someone has sat down and said "this is the way it should have been done") means that the beginner will be out on a limb when it comes to finding off-the-shelf programs to use. However, the many features it offers will undoubtedly prove attractive to some. It really is a shame; there are some excellent ideas in the Atom but spoilt by some basically unsound thinking at the system level as to which way computers are going in general and how to implement the ideas. I also think you should look closely at just what is offered by the various kits and options as it all sounds a bit rosy at times when you read the brochures.

My thanks to Acorn's Chris Turner for the supply of the machine and for answering all my questions.

TECHNICAL SPEC.

CPU	6502
Video Display	32 x 16, comprehensive graphics with expanded colour option promised.
RAM	2k minimum, 12k maximum
ROM	8k minimum, 4k floating point, room for another 4k
COMMS	cassette interface, can be used in Cambridge Ring
KEYBOARD	60 keys, direct cursor control, Copy key
BUS	Acorn's own 64-way Eurocard socket.
Disk	Not tested
Printer	Not tested

At a glance

FIRST IMPRESSIONS	
Looks	***
Setting Up	****
Ease of Use	*
HIGH LEVEL LANGUAGES	
Basic	**** (see text)
PERFORMANCE	
Processor	****
Cassette	***
Disk	N/A
Peripherals	N/A
EXPANDABILITY	
Memory	*
Cassettes	N/A
Disks	N/A
Bus	***
COMPATABILITY	
Hardware	**
Software	
Documentation	***
Value for Money	****

*****	excellent
****	v. good
***	good
**	fair
*	poor